Clustering Entity Relationship Diagrams: Enhancing Feedback Quality and Grading Consistency in Large Database Courses

Sohum Thadani*, Andrey Shor*, Soohong Ahn*, Lei Gong*, Abdussalam Alawini†, Hisham Benotman*
*Purdue University
{ssthadan, ahn70, gong148, hbenotma}@purdue.edu
†University of Illinois Urbana-Champaign
alawini@illinois.edu

***Abstract*—This innovative practice full paper introduces a tool for clustering Entity Relationship Diagrams (ERDs) and explores its application in large classes. ERDs are fundamental for database design in courses related to databases, data science, and software engineering. However, processing ERD homework submissions in large classes poses significant challenges due to the variety of design decisions made by students, leading to numerous diagram variations. This paper presents an ERD clustering tool designed to group similar ERD submissions, aiding instructors and teaching assistants in identifying popular solutions and common mistakes. The tool employs advanced object detection, OCR, and clustering technologies. We evaluated the tool using four datasets from two large public U.S. universities, with submissions ranging from 130 to 430 diagrams. Various clustering methodologies were assessed, highlighting the importance of incorporating ERD structure into the clustering process. Our findings indicate that the tool successfully generated adequate clusters, and that aiming for 10 clusters is appropriate regardless of the dataset size. The generated clusters included common approaches and mistakes, proving helpful for providing feedback and simplifying the grading process.**

***Index Terms*—Entity Relationship Diagrams, Diagram Clustering, Object Detection, OCR, K-means++, GMM, Educational Tools, Automated Grading, Feedback Generation.**

I. Introduction

Entity Relationship Diagrams (ERDs) are a fundamental concept for database design in database, data science, and software engineering courses. ERDs are used to represent data to be stored in an application database. For example, the ERD excerpt in Figure 1 represents data required by a car-rental application. An ERD may include one or more of the following components: entity sets (or entities for short), relationship sets (or relationships), weak entities, identifying relationships, and relationship attributes. Entities, such as 'Car' and 'Driver_rented' rectangles in Figure 1, are similar to tables in relational databases and classes in object-oriented programming languages. Weak entities, such as 'recalls' in Figure 1, are represented as double-edge rectangles. An entity or a weak entity rectangle includes a list of entity attributes, such as 'make' and 'model' in 'car'. One or more of those attributes are selected as a key for the entity. Those keys are underlined in Figure 1. Relationships, such as the 'Rented' diamond in Figure 1, connects two or more entities and/or weak entities. An identifying relationship, represented as a double-edge diamond, connects a weak entity to an entity. Relationships and identifying relationships have cardinalities on each side, such as the 0-1 on the 'recalls' side and the 0-many (the circle and chicken leg) on the 'car' side for the 'Recalled' identifying relationship. Relationships and identifying relationships may include attributes depicted as ovals in an ERD, such as 'start_date' and 'end_date' in the 'rented' relationship. Note that ERDs have a few different notations. We used a variation of the popular UML notation [1], where we represent relationship attributes as ovals instead

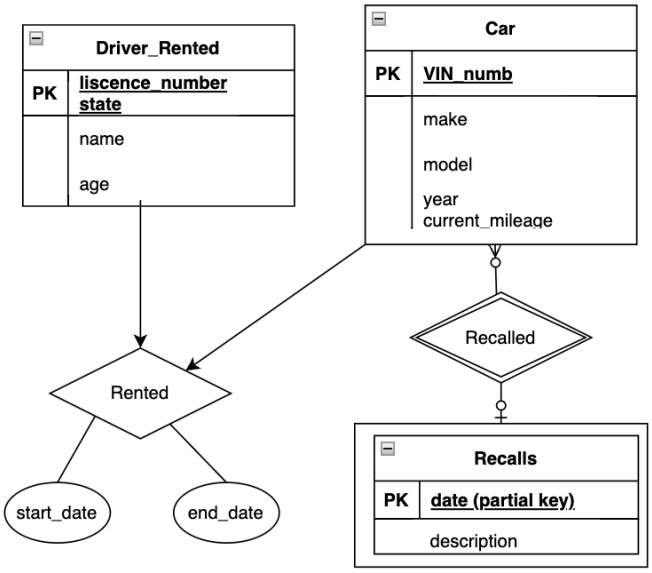

Fig. 1. Example ERD

An Entity-Relationship Diagram (ERD) consisting of several components: entities (rectangles), weak entities (double-edge rectangles), relationships (diamonds), identifying relationships (double-edge diamonds), and relationship attributes (ovals). Entities and weak entities include a list of attributes, such as 'make' and 'model'.

of rectangles. This change helps in object detection explained later.

Processing ERD homework submissions in large classes can be challenging. At Universities A and B, where we collected our datasets, the number of ERD submissions for one question reached 230 and 500 diagrams in the Spring 2024 semester, respectively. Additionally, tools for processing ERDs are limited and not well studied in the context of large classes. Providing detailed feedback is nearly impossible given the nature of ERDs, where students are required to make multiple design decisions, resulting in numerous diagram variations. Designing a grading rubric can also be challenging without examining a large number of submissions to identify the main solution variations. Despite appearing simple, an ERD assignment can be quite difficult for students. For example, at University A, the ERD assignment has had the lowest average score among six assignments in seven out of the last eight semesters.

We present an ERD clustering tool designed to group similar ERD submissions. This tool aids instructors and teaching assistants (TAs) in identifying popular solutions and common mistakes. By analyzing the clusters, instructors can provide detailed feedback by discussing these common solutions and mistakes with all students. The tool can also assist in designing a grading rubric by allowing instructors to examine a few representative diagrams from each cluster. Additionally, it can be used to order diagrams for grading, ensuring that similar diagrams from the same cluster are graded together, thereby enhancing grading consistency.

In our experiments, we assessed various clustering methodologies, including those relying solely on textual data extracted from ERDs, hybrid approaches combining textual data with quantitative metrics related to specific ERD components, and strategies employing multiple clustering algorithms. Using four datasets from two large public U.S. universities, we explored the use of such a tool in large classes with submissions ranging from 130 to 430 diagrams.

II. Tool Architecture

Our system comprises four key modules: an object detection module to detect ERD components (e.g., entities and weak entities), an OCR module for extracting text from ERD components, a module to convert each ERD to a vector, and a clustering algorithm to group similar ERDs. The primary input for the tool is a set of ERD diagrams for a particular assignment or exam question. Optionally, the text of the ERD question can also be provided to aid in text processing. The output of the clustering tool is a set of folders representing the generated clusters.

A. Object Detection Module

Our tool's object detection (OD) module harnesses the power of the YOLOv8 library [2], tasked with pinpointing the coordinates (i.e., bounding boxes) of ERD components within diagrams and classifying each component as an entity, weak entity, relationship, identifying relationship, or relationship attribute. We also tested multiple models in the TensorFlow object detection library, but YOLOv8 achieved better precision and recall in detecting ERD components. More details about object detection accuracy are provided in Section III-A.

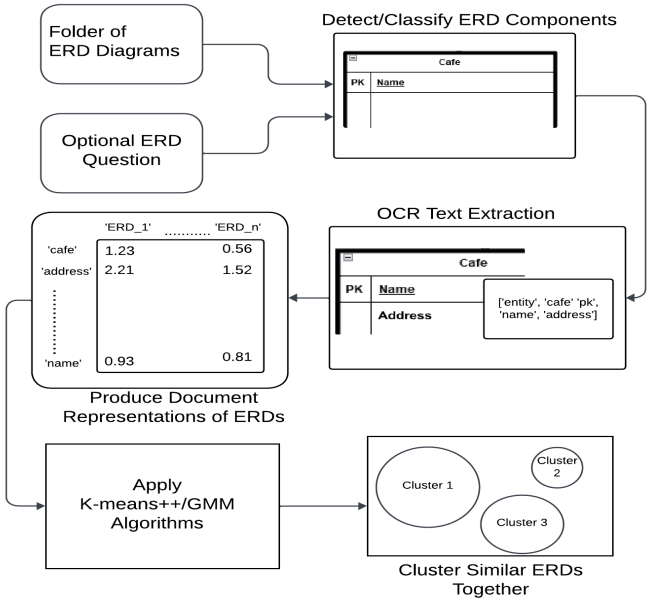

Fig. 2. ERD Clustering Tool System Design

The ERD clustering tool consists of six stages: inputting ERDs and an optional ERD question, detecting ERD components in each image, applying OCR to capture text from each ERD component, producing vectors for each ERD, clustering the ERDs, and finally generating the clusters stored in separate folders.

Our tool discards relationship cardinalities and lines in ERDs. Given their small size, detecting relationship cardinalities did not achieve acceptable accuracy. Additionally, lines do not have a regular shape suitable for training the model. However, our experiments show that these ERD components can be discarded for high-level clustering, as discussed later in Section III.

B. OCR Module

The Object Character Recognition (OCR) module extracts text from the bounding boxes generated by the object detection module. For example, in Figure II, we extract the text within the detected 'Cafe' entity and generate a list that includes the component type ('entity' in this example) and all extracted words. While evaluating potential OCR libraries, both PyTesseract and EasyOCR were thoroughly tested. EasyOCR was chosen for implementation due to its seamless integration within the tool's architecture and superior accuracy in recognizing text compared to PyTesseract.

C. Text Processing and vector generation:

The text generated by the OCR module is further processed using stemming and stop word removal. Stemming reduces words to their root form (e.g., "rented" and "renter" are both stemmed to "rent"), facilitating more effective comparison of

related terms when students use different word inflections. Additionally, stop word removal filters out common words such as "the" and "for," which do not contribute significantly to the semantic meaning of the text. This processed text provides a cleaner and more refined basis for subsequent analyses.

Another method utilized to process the extracted text from the OCR module was 'Edit Distance,' which was used to correct the text generated by the OCR module to match words in the ERD homework question. For instance, 'atrplane' returned by OCR can be corrected to 'airplane' (since 'airplane' is included in the question text). For the datasets used in this study, we asked students to use only words included in the question text to simplify text processing.

The third module converts the processed ERDs to vectors. The main component of these vectors is the frequency of each word in each diagram. Additional information, such as the frequency of various ERD components in each diagram, was later added to these vectors to produce various clustering methods. We refer to word frequencies and component frequencies as vector features.

D. Clustering and vector-feature configurations

We tested the following configurations of vector features and clustering algorithms.

1) **textBaseline:** This configuration included only ERD text features (i.e., word counts) without any information about the ERD structure (e.g., counts of entities and relationships). We used the K-means++ algorithm [3] to cluster the ERDs. This method serves as a baseline since it relies solely on simple document processing techniques (OCR output) without leveraging information we get from the object detection module.
2) **textCompFreq:** In addition to text-based features, this configuration adds the frequency of each ERD component type in a diagram (e.g., number of entities, number of weak entities, etc.). We used the K-means++ algorithm to cluster the ERDs.
3) **textPrefixCompFreq:** This configuration modifies textCompFreq by adding a component-type prefix to each word in an ERD. For instance, if ERD1 has 'age' as an entity attribute, it will have a frequency of 1 in the 'entity_age' feature, whereas ERD2 having 'age' as a relationship attribute will have a frequency of 1 in the 'rel_age' feature. This ensures that ERD1 and ERD2 are considered different because they contain the word 'age' in different components. For this configuration, we used the K-means++ algorithm to cluster the ERDs.
4) **twoClusterings:** This configuration uses the same vector features as textPrefixCompFreq but utilizes both the K-means++ and the GMM clustering algorithms. We use the K-means++ algorithm to generate initial clusters and then examine them for long and thin clusters. If such clusters are found, we recluster the ERDs using the GMM algorithm, which is better suited for handling long and thin clusters compared to K-means++.

III. Evaluation

A. Object Detection and OCR Evaluation

We evaluated the object detection (OD) module using the F1 score, which is computed as the harmonic mean of precision and recall. Precision reflects the module's ability to minimize false-positive detections, ensuring accurate identification of ERD components, while recall measures its effectiveness in avoiding overlooked components within the ERDs. The F1 measure ranges between 0 and 1, with 1 indicating perfect detection of all ERD components. We used a testing dataset of 10 ERDs from 7 different questions. The questions were in the domains of video games, airline reservations, taxis, garage spaces, Formula 1 racing, dealerships, and TV series. The ground truth used to compute the F1 measure included all component types and text in the 10 ERDs.

The object detection module exhibited an average F1 score of 0.96 (96%) on the test dataset, indicating a high accuracy in detecting ERD components. The OD module achieved perfect detection of ERD objects in 8 out of the 10 ERDs within the test set, with a minimum F1 score of 0.89 (89%). It is worth noting that the detection of relationship cardinalities did not achieve acceptable accuracy and was therefore discarded in our tool. Further analysis revealed that cardinalities are not crucial for high-level clustering (e.g., grouping the ERDs into 10 or 20 clusters). However, cardinalities might be important for more detailed clustering.

We also tested the OCR module using the same dataset. Similar to the OD evaluation, the OCR module's performance was assessed using the F1 score. The OCR module achieved average and median F1 scores of 0.876 and 0.91, respectively. Note that the OCR module relies on the object detection module to recognize ERD components.

In summary, the combination of YOLOv8 for object detection and EasyOCR for text recognition achieved high accuracy in most ERDs. TensorFlow object detection and PyTesseract OCR libraries achieved lower F1 score values.

B. Clustering Evaluation

We started by evaluating the first three clustering and vector-features configurations: textBaseline, textCompFreq, and textPrefixCompFreq, described in Section II-D. All configurations used the K-means++ clustering algorithm but differed in the vector features used to represent the ERD submissions. We used 12 datasets from 4 different questions, each dataset containing 10 to 14 ERDs.

To create a ground-truth clustering, we asked a database instructor to manually cluster the ERDs in each dataset. The instructor determined an appropriate number of clusters K for each dataset, which ranged from 2 to 5, with an average K of 4. We then used the Rand Index [4] to compare the clusters generated by the tool to those created manually by the database instructor.

The Rand Index [4] measures the similarity between two different ways of grouping items by comparing pairs of items/ERDs and determining if they are consistently grouped

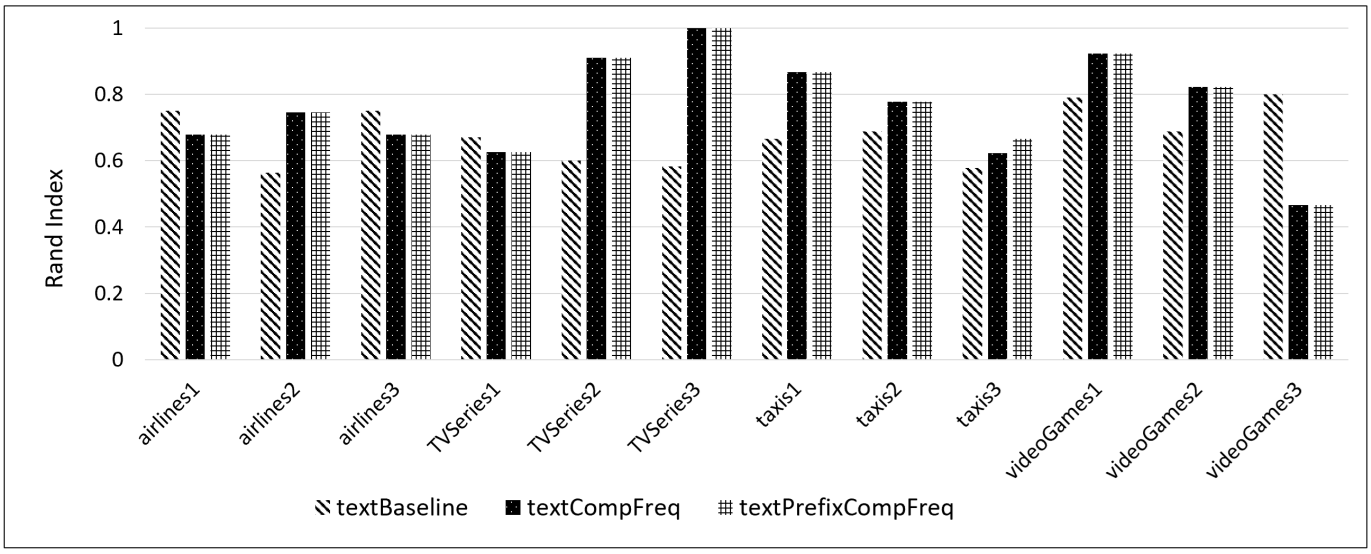

Fig. 3. The Rand index scores of three clustering configurations in twelve datasets.

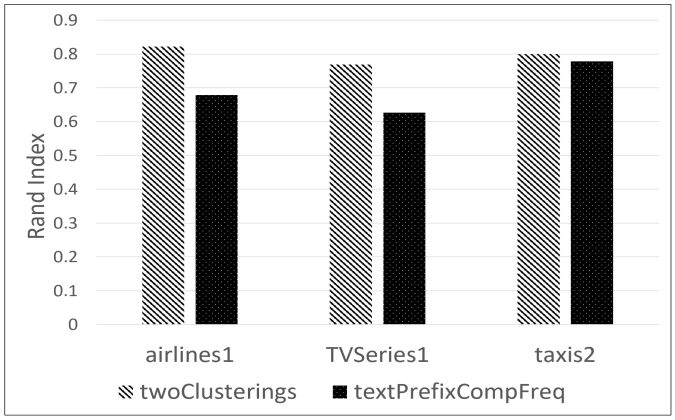

Fig. 4. The Rand index scores for two tool configurations. While textPrefixCompFreq uses K-Means++, the twoClusterings configuration dyanmically switches from K-Means++ to GMM clustering algorithm as in these three datasets increasing the overall rand index.

together or apart. The Rand Index value lies between 0 and 1, with a higher value indicating greater agreement between the two clusterings and a lower value suggesting more disagreement.

On average, the textCompFreq and textPrefixCompFreq configurations outperformed the textBaseline configuration, as shown in Figure 3. The average Rand Index scores for textBaseline, textCompFreq, and textPrefixCompFreq were 0.677, 0.760, and 0.764, respectively. The textCompFreq and textPrefixCompFreq configurations scored higher than textBaseline in 8 out of 12 datasets, while textBaseline was better in four datasets. Our findings indicate that using only ERD text may generate an adequate clustering. However, including the frequencies of each ERD component type improves the clustering accuracy, highlighting the importance of considering the structure of ERDs in addition to the text.

Comparing textPrefixCompFreq and textCompFreq, we observed a negligible positive effect from the inclusion of ERD-component type as word prefixes. This latter finding indicates that similar words in multiple ERDs tend to be located in the same component type (e.g., if 'car' is an entity attribute in ERD1, then if 'car' is in ERD2, it is likely to be an entity attribute as well).

In four datasets, textBaseline was better than the other two configurations. We observed cases where textCompFreq and textPrefixCompFreq placed diagrams with almost the same structure in different clusters because of slight differences in component-type frequencies. For example, if ERD1 has the same structure as ERD2 but changes one entity to a weak entity, this small difference might result in placing the two diagrams in different clusters in textCompFreq and textPrefixCompFreq. In manual clustering, such small differences were often overlooked, especially when larger differences existed within the dataset, resulting in these diagrams being placed in the same cluster.

Finally, we compare textPrefixCompFreq (the best configuration so far), which relies on the K-Means++ clustering algorithm, to the twoClusterings configuration, which dynamically selects between K-Means++ and the GMM clustering algorithm. K-Means++ used by textPrefixCompFreq works best on datasets with spherical clusters. However, some datasets consist of non-spherical clusters because ERDs may contain different amounts of variation along certain features. For instance, features related to word frequencies have relatively less variation compared to ERD-component frequencies.

To address these issues, we applied GMM clustering to datasets that our algorithm evaluated as having more non-spherical clusters. GMM clustering is well-suited for clustering datasets with long and thin clusters or datasets containing

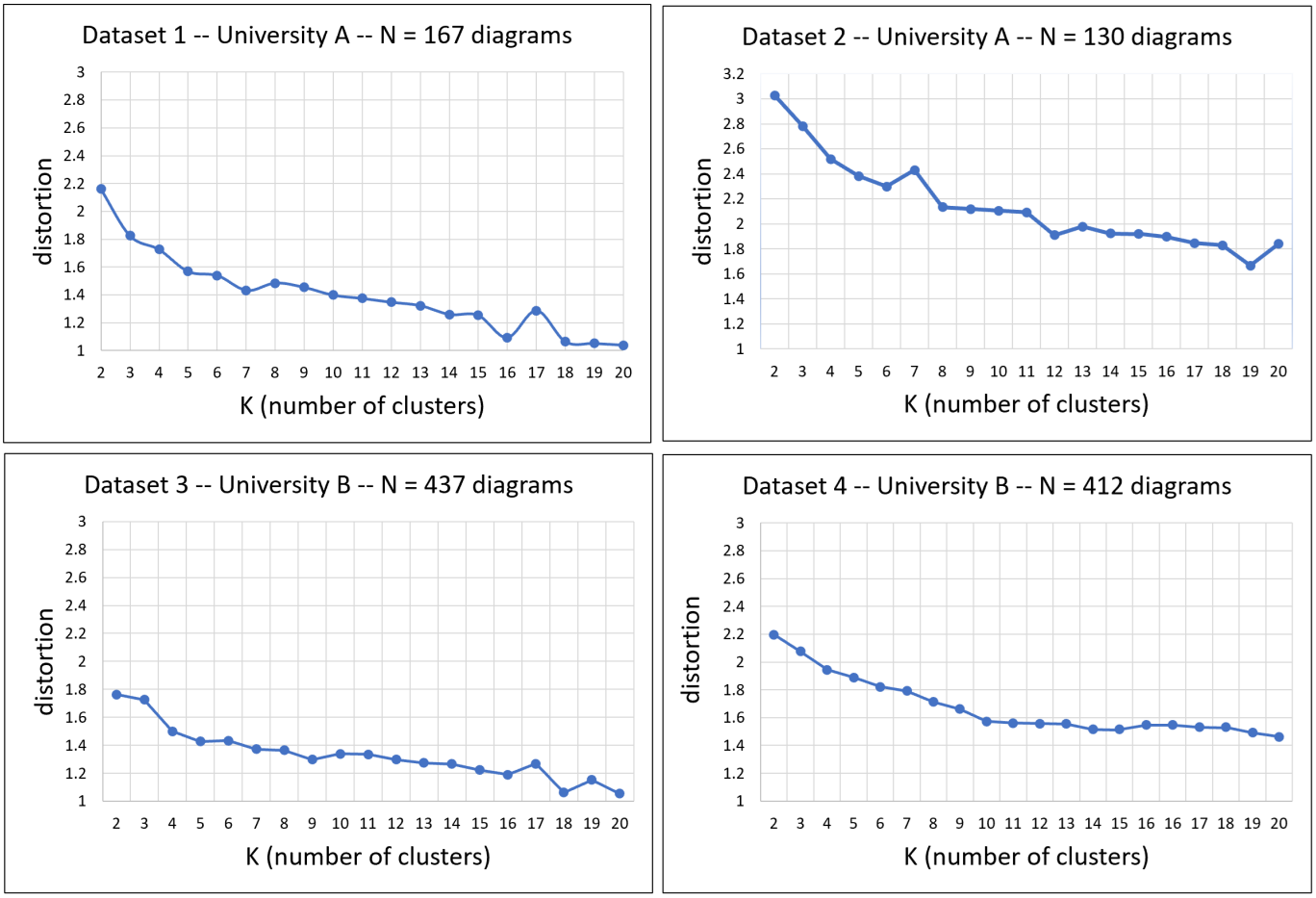

Fig. 5. The elbow method curve for the four datasets. Lower distortion values represent more homogeneous clusters. Aiming for ten clusters seems to work well across all datasets, regardless of their size. This approach helps achieve homogeneous clusters while also limiting the number of clusters, making manual analysis easier. Dataset 2 included the most challenging question, resulting in more ERD variations and higher distortion values.

clusters with significantly more variation along certain feature spaces. This dynamic selection helps ensure that the clustering method used is best suited to the specific characteristics of the dataset, potentially improving clustering accuracy and relevance.

The use of the twoClusterings configuration increased the average Rand Index score from 0.764 (in textPrefixCompFreq) to 0.793. However, we found that our tool switched to GMM in only three datasets. In these three datasets, GMM indeed performed better than K-Means++, as shown in Figure 4.

C. Clustering Large Datasets

To test the effectiveness of the clustering tool in large classrooms, we used four datasets. Datasets 1 and 2 are from University A, with 167 and 130 diagrams, respectively. Datasets 3 and 4 are from University B, with 437 and 412 diagrams, respectively. Datasets 1 and 2 were collected in different semesters, while Datasets 3 and 4 were collected in the same semester (two questions in the same assignment).

The optimal K value (number of clusters) seems to be between 7 and 10, consistent across most datasets regardless of their size. To find the best K values, we computed the elbow method curve, shown in Figure 5. For each K value between 2 and 20, we ran the clustering algorithm using the twoClustering configuration to compute distortion, which is shown on the Y-axis. Lower distortion indicates lower heterogeneity within each cluster. The elbow method helps identify a small K value that still yields low distortion (shown as the elbow point in the graph). We noticed that K values between 7 and 10 work well for all datasets regardless of their size.

Distortion (i.e., average homogeneity for all clusters) was consistent in Datasets 1, 3, and 4, with values between 1.2 and 1.6. However, Dataset 2 had a higher value of 2.13 due to the complexity of its ERD question, resulting in more diverse student submissions. For Dataset 2, a larger K value might return more homogeneous clusters. However, based on analyzing these four datasets, we believe that using K = 10 as a default value is expected to return acceptable clusters. Additionally, manually examining a few diagrams from each of the 10 clusters is manageable for instructors and TAs.

Performance-wise, the clustering tool can process a dataset of 150 diagrams in about 15 to 20 minutes when using a

shared university cluster powered by an Nvidia Tesla V100 GPU. The same task takes about 90 minutes on a laptop with an M1 chip. GPU capability is required for the object detection phase. The processing time is reasonable given that ERD clustering is likely to be used offline to analyze students' submissions in assignments and exams. Using the tool online, such as processing ERD images during a lecture, would be challenging for large datasets unless more powerful hardware is available.

D. Use Case Analysis: Dataset 1

We discuss our observations using Dataset 1, which consists of 167 diagrams, as an example. The ten generated clusters included 15, 5, 18, 11, 13, 21, 23, 25, 7, and 29 diagrams for clusters 1 through 10, respectively.

Cluster 1 consisted of 15 diagrams. All diagrams had 5 entities and the same number of relationships, with no relationship attributes. Most diagrams (13 out of 15) did not include any weak entities or identifying relationships. Two entities, 'region' and 'developer', were exactly the same across all diagrams. One entity consistently included the same attributes in all diagrams, with the only difference being the key attributes. The 'platform' and 'video games' entities were mostly the same in all diagrams, except that some diagrams incorrectly included foreign-key attributes in these entities. Foreign key attributes are essential in the popular relational database model. Including foreign-key attributes in ERDs is a common mistake observed in most datasets.

Although the tool did not recognize links, 14 out of 15 diagrams in Cluster 1 had the same structure, with consistent connections among the entities and relationships. This is an example of how ERDs with the same components tend to have similar connections among those components. However, there are exceptions, especially with n-way relationships that are not recognized by our tool.

In the context of building a grading rubric or assessing diagrams in the same cluster together, most of the diagrams in Cluster 1 are likely to receive a similar grade with slight variations based on the few differences of weak entities and the few extra attributes.

Clusters 2, 6, and 8 had a total of 51 diagrams. Of these, 47 diagrams closely matched the key solution. All 47 diagrams in these clusters had four entities and one relationship attribute (compared to 5 entities in Cluster 1). All diagrams had an identical structure with consistent connections among ERD components. The four incorrect diagrams included weak entities or incorrect connections. Clusters 2, 6, and 8 could have been merged into one cluster, as the only differences between them were some word choices. Except of the 4 outliers, most of the diagrams in Clusters 2, 6, and 8 are close to the key solution and could be easily awarded a full score.

Clusters 3, 5, and 7, with 18, 13, and 23 diagrams, respectively, included diagrams with some weak entities and/or identifying relationships. Cluster 3 consistently included four entities and one weak entity (the game sales entity), with slight differences mostly in cardinalities. Cluster 5 included diagrams with 2 or 3 weak entities and/or 2 or 3 identifying relationships, highlighting students who incorrectly and heavily relied on weak entities to model an application's data requirements. In Cluster 7, the most dominant feature was having one identifying relationship, with or without weak entities (in 14 out of 23 ERDs). These clusters highlight the common mistake of relying (sometimes heavily) on weak entities and identifying relationships. A popular mistake that we found in Cluster 7 and other datasets as well is having an identifying relationship without a weak entity (an identifying relationship must connect a weak entity to an entity). These mistakes can be discussed with students in subsequent lectures. In terms of grading, Cluster 5 will be the most challenging since component-frequency features were essential in creating this cluster. However, diagrams were sometimes different even though they included similar numbers of weak entities and identifying relationships.

Clusters 9 and 10 included incorrect submissions that did not follow the submission rules. Five out of seven diagrams in Cluster 9 included an assumptions or a description box. Cluster 10 included 29 diagrams that used low-resolution images, had large amounts of text around the diagram, or included colors. Object detection and OCR did not work well with these images, causing those diagrams to be clustered together. We observed a similar cluster in each dataset, comprising between 12% and 16% of the overall dataset size. This cluster highlights the importance of reminding students about the required ERD format, such as using high-resolution images and avoiding colors.

Datasets 3 and 4 were quite similar to Dataset 1, with a number of main approaches (solutions) ranging between 3 and 7, along with some variations. Dataset 2 included a larger number of variations, indicating a more complex question.

For all datasets, our manual clustering and the analysis of the clusters generated by the tool suggest that certain ERD features related to ERD structure should be prioritized in the clustering process. For example, the number of ERD components and entity names should be prioritized over entity-attribute names, relationship-attribute names, and relationship cardinalities. Attribute names often acted as noise and negatively impacted the clustering in some cases. A future tool could test the idea of minimizing the effect of these less-important features. We expect that a subset of the features may work well for high-level clustering that generates 5 to 20 clusters for a large dataset.

IV. Related Work

A closely related project is the work of Thomas et al. [5], who presented an autograding tool for ERDs. However, the notation they used is different from the UML notation we use in this study, as their notation did not include entity and relationship attributes. The closest to our work is the tool developed by Dahanayake et al. [6], who processed EER images. The EER notation they processed is very different from the UML notation we used in this study. Their overall tool architecture is similar to ours, but they used TensorFlow

and PyTesseract, which we tested and found to be either less accurate or having more processing overhead compared to YOLOv8 and EasyOCR. The tool developed by Dahanayake et al. [6] processed EER images for cheating detection. Our study focuses on clustering to facilitate feedback and grading and to study patterns in ERD submissions of large classes. Other studies have worked on plagiarism detection in other types of diagrams, such as in flowcharts [7] [8].

A body of research has addressed the detection of plagiarism in scientific figures included in research papers. Meuschke et al. [9] utilized OCR to extract text from figures, complementing this with other image features to identify potential plagiarism. Eisa et al. [10] focused specifically on detecting plagiarism by analyzing textual features associated with figures, such as figure captions and the text referencing the figures.

Several projects have processed UML diagrams for various purposes [11] [12] [13]. Ma et al. [11] clustered UML diagrams to enhance UML retrieval from repositories for software reuse. Yuan and Ma [12] further classified UML diagrams. The work most closely related to ours is the prototype presented by Huber and Hagel [13], which processes UML diagrams using YOLO and OCR to provide immediate feedback to students.

Several projects have clustered and/or analyzed various types of student submissions in large classes. Lokkila et al. [14] clustered code submissions in an introductory programming class of 467 students to identify those who were struggling. In the context of database courses, multiple researchers have focused on analyzing SQL queries. Similar to this study, Weston et al. [15] introduced a SQL-query clustering tool and reported its use for clustering submissions in large classes. Ahadi et al. [16] analyzed a large dataset of SQL queries collected over eight years to identify common mistakes. Chang and Forbus [17] had a similar research goal of finding common approaches in students' diagram submissions. However, their dataset was small (28 students), and the diagrams were different, as the study clustered hand-drawn sketches from an undergraduate geoscience course.

To the best of our knowledge, no other studies have clustered ERD submissions or explored using an ERD clustering tool in the context of large classes.

A. Conclusion

This study presents an Entity Relationship Diagram (ERD) clustering tool designed to aid instructors and teaching assistants in analyzing and grading large volumes of ERD submissions in image format. The tool leverages advanced object detection, OCR, and clustering technologies to cluster ERD submissions. The tool's ability to group similar ERDs together allows for more targeted feedback, identification of common solutions and mistakes, and streamlined rubric development. The evaluation of various clustering methodologies highlighted the importance of incorporating ERD structure into the clustering process. The tool's performance on different computing resources indicates its feasibility for practical use in educational settings.

Our experiments, conducted with datasets from two large public U.S. universities, demonstrate the tool's effectiveness in clustering ERDs based on their structural components and textual data. We found that setting the desired number of clusters between 7 and 10 was suitable for all datasets, regardless of their size. Most of the generated clusters included common approaches and/or mistakes, which were helpful for providing feedback and simplifying the grading process.

B. Future Work

While the current implementation of the ERD clustering tool shows promising results, several areas for future improvement and research remain. Future versions of the tool could explore the impact of different feature selection techniques to minimize the influence of less important features, such as entity-attribute names, and relationship-attribute names, which sometimes act as noise in the clustering process.

Developing capabilities for real-time processing of ERDs during lectures or interactive sessions could further enhance the tool's utility. This would require optimizing the tool for lower-latency environments and potentially incorporating more powerful hardware.

Conducting further studies to assess the long-term impact of using the clustering tool on student performance and learning outcomes would provide valuable insights into its educational effectiveness.

REFERENCES

[1] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts*. McGraw-Hill, 2019.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[3] D. Arthur, S. Vassilvitskii *et al.*, "k-means++: The advantages of careful seeding," in *Soda*, vol. 7, 2007, pp. 1027–1035.

[4] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.

[5] P. Thomas, N. Smith, and K. Waugh, "Automatically assessing graph-based diagrams," *Learning, Media and Technology*, vol. 33, no. 3, pp. 249–267, 2008.

[6] H. Dahanayake, D. Samarajeewa, A. Jayathilake, D. Bandara, A. Karunasena, and L. Weerasinghe, "Plagiarism detection tool for enhanced entity-relationship diagrams," in *2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE, 2021, pp. 0598–0606.

[7] J. S. Kuruvila, M. L. VL, R. Roy, T. Baby, S. Jamal, and K. Sherly, "Flowchart plagiarism detection system: an image processing approach," *Procedia computer science*, vol. 115, pp. 533–540, 2017.

[8] S. Arrish, F. N. Afif, A. Maidorawa, and N. Salim, "Shape-based plagiarism detection for flowchart figures in texts," *arXiv preprint arXiv:1403.2871*, 2014.

[9] N. Meuschke, C. Gondek, D. Seebacher, C. Breitinger, D. Keim, and B. Gipp, "An adaptive image-based plagiarism detection approach," in *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries*, 2018, pp. 131–140.

[10] T. A. E. Eisa, N. Salim, and S. Alzahrani, "Figure plagiarism detection based on textual features representation," in *2017 6th ICT International Student Project Conference (ICT-ISPC)*. IEEE, 2017, pp. 1–4.

[11] Z. Ma, Z. Yuan, and L. Yan, "Two-level clustering of uml class diagrams based on semantics and structure," *Information and Software Technology*, vol. 130, p. 106456, 2021.

[12] Z. Yuan and Z. Ma, "Supervised classification of uml class diagrams based on f-knb." *International Journal of Software Engineering & Knowledge Engineering*, vol. 33, no. 8, 2023.

[13] F. Huber and G. Hagel, "Work-in-progress: Towards detection and syntactical analysis in uml class diagrams for software engineering education," in *2020 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 2020, pp. 3–7.

[14] E. Lokkila, A. Christopoulos, and M.-J. Laakso, "A clustering method to detect disengaged students from their code submission history," in *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1*, 2022, pp. 228–234.

[15] M. Weston, H. Sun, G. L. Herman, H. Benotman, and A. Alawini, "Echelon: An ai tool for clustering student-written sql queries," in *2021 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2021, pp. 1–8.

[16] A. Ahadi, V. Behbood, A. Vihavainen, J. Prior, and R. Lister, "Students' syntactic mistakes in writing seven different types of sql queries and its application to predicting students' success," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 2016, pp. 401–406.

[17] M. Chang and K. Forbus, "Clustering hand-drawn sketches via analogical generalization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 27, no. 2, 2013, pp. 1507–1512.